REMARKS

Claims 10-26 are pending in the present application. No amendments to the claims are made by this Response. Reconsideration of the claims is respectfully requested.

I. Response to Examiner's Arguments

The Final Office Action maintains the rejection of claims 10-13 and 16-26 under 35 U.S.C. § 103(a) based on Gong (U.S. Patent No. 6,192,476) in view of Bak et al. (U.S. Patent No. 6,009,517) and the rejection of claims 14-15 under 35 U.S.C. § 103(a) based on Gong in view of Bak et al. and further in view of the Capabilities Classes reference. These rejections are traversed for at least the same reasons as set forth in the Response filed July 23, 2003, the remarks of which are hereby incorporated by reference. Since the rejections are identical to the rejections in the previous Office Action, the following remarks will only address the Examiner's statements with regard to Applicants' arguments appearing on pages 5-7 of the Final Office Action.

As discussed in the July 23, 2003 Response, Gong is directed to a system for controlling access to computer resources. The system of Gong is directed to determining whether a thread may access a particular resource. With the system of Gong, a thread may access a particular resource only if at least one permission of a protection domain associated with each method in the call hierarchy of the thread encompasses the permission required to perform the requested action. An exception is provided in the form of a privilege flag that may be associated with a method in the call hierarchy.

Gong does teach that the call hierarchy, or call stack, is comprised of frames and that the Check Permission method traverses the call stack and determines if each method in the call stack enables the privilege (see Figure 4 of Gong). However, as argued in the July 23, 2003 Response, Gong does not teach or suggest using a thread identifier to locate a linked list or searching the linked list for an entry having a stack frame pointer that matches the stack frame pointer of the method for which a privilege is being provided.

In addition, Applicants argued that Bak does not provide for the deficiencies of Gong. Bak is directed to a system that enables frames for code written in different languages to be added to the same call stack. As illustrated in Figure 5 of Bak, a prior art call stack includes frames that have a first portion that identifies a return address, a second portion that identifies state variables, local variables and operands, and a third portion that includes a frame pointer to point to the next frame in the call stack. With the system of Bak, as illustrated in Figure 7, the call stack includes Java frames, and a block of data identifying an entry frame of the call stack, the last java stack pointer and the last java frame pointer that point to the next Java frame in the call stack. A frame of a different type may be placed between Java frames as illustrated. Thus, similar to the frame pointers in Figure 5, the block in Figure 7 allows the call stack to be traversed with regard to Java frames. However, as with Gong above, there is no teaching or suggestion in Bak to identify a linked list using a thread identifier or searching a linked list of stack frame extensions for an entry having a stack frame pointer that matches the stack frame pointer of the method.

In response to these arguments, the Final Office Action alleges:

> Firstly, Gong prior art reference as Applicant would agree discloses a stack that is made up of stack frames of methods and the methods contain privileges. As the Applicant rightfully noted the Gong prior art reference traverses the stack each time a request is received to determine if the method in the frame of stack provides a privilege. The act of traversing is the same as the act of searching of the claimed limitation because both the Gong reference and the claimed limitation searches for a method that contains a privilege. Although, the Gong reference is silent with respect to matching a stack frame pointers in order to locate the method that contain a privilege this limitation is inherent in traversing/searching a stack. This is because each frame of a stack inherently includes a stack frame pointer that links or pointer to the next frame. And in traversing/searching the stack the stack frame pointer of each frame is examined in order determine the next frame and subsequently the frame/method that contains a privilege.
>
> This notwithstanding the office action introduced Bak prior art reference to show the traversing/searching of stack using stack frame pointer. As Applicant rightfully agreed (page 11 lines 8-12) stack frame pointer may used to traverse/search a stack for a particular frame. In using the stack frame pointer the frame/method that contains a privilege could be located.

(pages 5-6 of the Final Office Action; emphasis added)

It is believed that the fundamental error in the Examiner's interpretation of the present claims reading on the teachings of the Gong and Bak references is that the Examiner has not recognized that the linked list in the claim is not the call stack but rather, a linked list of stack frame extensions separate from the call stack. Claim 10 clearly recites:

> 10. A process for providing a privilege for a method of a current thread that is currently executing in a run-time environment in a data processing system, the run-time environment having a stack comprising stack frames with stack frame pointers for associated methods, the process comprising the computer-implemented steps of:
>
> using a thread identifier of the current thread, locating a linked list; and
>
> searching the linked list for an entry having a stack frame pointer that matches the stack frame pointer of the method, wherein an entry of the linked list is a stack frame extension. (emphasis added)

It is clear from the preamble of claim 10 that the runtime environment has a stack comprising stack frames with stack frame pointers for associated methods. The body of the claim recites locating a linked list, not locating the stack. Because both a "stack" and a "linked list" are recited in claim 10, it is clear that these are two different types of data structures. In other words, the linked list is not the stack and the stack is not the linked list. If these were the same data structure, rather than reciting "locating a linked list", Applicants would have recited "locating the stack."

Furthermore, claim 10 clearly recites that entries in the linked list are stack frame extensions. Claim 10 does not recite that the entries in the linked list are stack frames, such as would be found in the call stack, but stack frame extensions. Therefore, it is clear from the text of claim 10 that there are two data structures – the stack and the linked list. The linked list is not the stack and the stack is not the linked list.

Therefore, the Examiner's interpretation that the linked list is the call stack and that the cited references, which only discuss traversing the call stack, teach the features of claim 10, is incorrect. That is, the Examiner's allegations that traversing/searching the stack in Gong and Bak is the same as the searching recited in claim 10 is incorrect at least

because the searching performed in claim 10 is with respect to a linked list of stack frame extensions located using a thread identifier of a current thread, not the call stack as in Gong and Bak. Claim 10 does not recite searching the call stack. Claim 10 recites locating a linked list having entries that are stack frame extensions, using a thread identifier of a current thread, and then searching the linked list for an entry having a stack frame pointer that matches the stack frame pointer of the method. These features are neither taught nor suggested by either Gong, Bak, or any alleged combination of Gong and Bak.

Furthermore, nowhere in Gong or Bak is there any teaching or suggestion of searching the linked list for an entry having a stack frame pointer that matches the stack frame pointer of the method. The Final Office Action admits that Gong is silent with respect to this feature. However, based on the erroneous interpretation that the linked list in claim 10 is the call stack, the Examiner alleges that this feature is inherent simply because frames in a stack inherently include stack frame pointers.

Applicants agree that stack frames in a call stack have stack frame pointers. In fact, this is explicitly stated in the preamble of claim 10. The preamble of claim 10 recites that the runtime environment has a stack comprising stack frames with stack frame pointers for associated methods. However, the body of the claim is not operating on the stack. To the contrary, the body of claim 10 recites a linked list of stack frame extensions. The operations of locating a linked list and searching the linked list are not performed on the call stack, as previously discussed above. Thus, the searching of a linked list for an entry having a stack frame pointer that matches the stack frame pointer of the method does not equate to simply searching a call stack for a stack frame. To the contrary, this feature of claim 10 is stating that the linked list of stack frame extensions is searched to find an entry in the linked list that has a stack frame pointer that matches the stack frame pointer that is in the stack for the method of the current thread. Such a feature is not taught or suggested by either Gong or Bak.

With regard to Applicants' argument that neither Gong nor Bak teach a linked list of stack frame extensions, the Examiner alleges:

> A stack is made up finite number of frame extensions and these frames contain method entries and are linked together by a stack frame pointer and thus, a linked list of method entries/stack frame extension.

Applicants respectfully disagree. A stack is made up of entries that are stack frames. A stack does not include stack frame extensions, despite the allegations made by the Examiner. Thus, the call stacks in Gong and Bak do not constitute a linked list of stack frame extensions. Furthermore, as noted above, claim 10 calls for both a stack and a linked list having entries that are stack frame extensions and thus, the linked list is not the same as the call stack.

With regard to Applicants' argument that neither Gong nor Bak teaches or suggests locating a linked list using a thread identifier of a current thread, the Examiner alleges:

> A thread inherently includes an identifier, that uniquely identify a thread and the thread 306 is no different. According the Gong prior art reference every call stack has an associated thread (Col. 10 Ln. 61-67). In associating a thread with its call stack some kind of an identifier is needed to tie/link the thread to the call stack.
>
> Since each frame in stack is a stack frame extension and has a stack frame pointer as explained above and as Applicant agrees that Gong teaches the use of privilege and validation information the argument about Gong not disclosing privilege and validation information in a stack frame extension along with stack frame pointer is not valid.

While threads have thread identifiers, there is nothing in Gong or Bak that teaches to locate a linked list of stack frame extensions based on a thread identifier. Again the Examiner is misinterpreting the claim as reading on the call stack. Applicants have shown above how the linked list is not the call stack but a linked list having entries that are stack frame extensions. The stack frames of a call stack are not stack frame extensions (see Figure 5 of the present application, for example). Thus, the Examiner's continued insistence that the call stack includes stack frame extensions is simply incorrect. The call stack includes stack frames, not stack frame extensions.

Therefore, even if the Examiner's statements are correct, and it is possible to identify a call stack based on a thread identifier, this still does not obviate the claimed

features because the claims are not directed to locating the call stack based on a thread identifier. To the contrary, the claims are directed to locating a linked list of stack frame extensions based on a thread identifier of a current thread. As discussed in detail above, from the text of the claim it is clear that the linked list is not the stack, despite the allegations made by the Examiner in the Final Office Action.

Thus, in view of the above, Applicants respectfully submit that neither Gong nor Bak, either alone or in combination, teach or suggest the features of independent claim 10. Independent claims 19 and 26 recite similar features to those in claim 10 but in system and computer program product format, respectively. Therefore, claims 19 and 26 are considered to define over the alleged combination of Gong and Bak for similar reasons as noted above.

With regard to independent claim 18, the Final Office Action still fails to address the specific features set forth in claim 18 and the Examiner's arguments do not address Applicants arguments with regard to claim 18.

As stated in the previously filed Response, claim 18 contains different features from those in claim 10 – features not addressed by the rejection of claim 10. For example, claim 18 recites "a set of stack frame extensions, wherein a stack frame extension comprises: a pointer to a stack frame for a method; a data field for privilege data for the method; a data field for validation data for the method." None of these fields of a stack frame extension are addressed by the Final Office Action and there is no showing where either of Gong or Bak allegedly teach these features. This is because there is no teaching or suggestion in Gong or Bak with regard to these fields of a stack frame extension. Since the Final Office Action fails to even address these features, the Final Office Action has failed to establish a prima facie case of obviousness with regard to these features and the rejection of claim 18 should be withdrawn.

In view of the above, Applicants respectfully submit that neither Gong nor Bak, either alone or in combination, teach or suggest the features of independent claims 10, 18, 19 and 26. At least by virtue of their dependency on claims 10 and 19, respectively, neither Gong nor Bak, either alone or in combination, teach or suggest the features of dependent claims 11-13, 16-17 and 20-25. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 10-13 and 16-26 under 35 U.S.C. § 103(a).

Furthermore, neither Gong nor Bak, either alone or in combination, teach or suggest the specific features recited in dependent claims 11-13, 16-17 and 20-25. Applicants offered arguments with regard to the features of each of claims 11-13, 16-17 and 20-25 and illustrated how the Gong and Bak references do not teach the features of these claims. The Examiner's response to Applicants' arguments does not address any of these arguments with regard to the dependent claims. Thus, the Examiner has not shown where or why Applicants' arguments with regard to these dependent claims are not persuasive. Therefore, Applicants respectfully submit that, for the reasons set forth in the Response filed July 23, 2003, neither Gong nor Bak, either alone or in combination, teach or suggest the features set forth in dependent claims 11-13, 16-17 and 20-25. Therefore, the rejection of these claims should be withdrawn.

With regard to claims 14 and 15, Applicants argued the rejection of these claims based on the same reasoning set forth with regard to independent claim 10. Furthermore, it was argued that the additional reference, Introduction to the Capabilities Classes (ICC), does not provide those features that are deficient in the Gong and Bak references. Therefore, any alleged combination of Gong, Bak and ICC still would not render obvious the invention recited in claims 10, 14 and 15. The Examiner has not offered any rebuttal of this position other than that noted above with regard to claim 10. Therefore, for the reasons set forth in the July 23, 2003 Response, and those set forth above with regard to claim 10, Applicants respectfully submit that the rejection of claims 14 and 15 should be withdrawn.

II. Conclusion

It is respectfully urged that the subject application is patentable over the Gong, Bak and ICC references, and is now in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

Respectfully submitted,

DATE: January 15, 2004

Stephen J. Walder, Jr.
Reg. No. 41,534
Carstens, Yee & Cahoon, LLP
P.O. Box 802334
Dallas, TX 75380
(972) 367-2001
Attorney for Applicants